

UNIVERSITY OF OTTAWA
FACULTY OF ENGINEERING
SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

ITI1100

Laboratory Experiments

A. Karmouch, V. Groza and A. Al-Dhaher

Winter 2016

TABLE OF CONTENTS

Table of Contents	i
1 - Laboratory instructions	1
1.1 - Laboratory rules.....	1
1.2 - Laboratory REPORT FORMAT	2
2 - General Introduction.....	3
2.1 - Digital Test Equipment.....	3
2.2 - Combinatorial Logic Circuits	3
3 - Overview of the Altera-DE2-115	6
3.1 - Layout.....	6
3.2 - Physical characteristics.....	6
3.3 - Quick debugging check list	7
4 - Introduction to quartus II.....	8
4.1 - Downloading and installing quartus	8
4.2 - Creating a new project.....	9
4.3 - Creating a new schematic.....	10
4.4 - Compiling the project	14
4.5 - Simulating a circuit.....	15
4.6 - Pin assignment.....	20
4.7 - Loading project on the card.....	21
4.8 - Using 7 segment display.....	24
4.9 - Creating a symbol from a circuit	25
5 - Laboratory experiments.....	26
5.1 - Lab 1 - Logic Gates	26
5.2 - Lab 2 - Boolean Logic.....	31
5.3 - Lab 3 - Decoders, Displays and Multiplexers	36
5.4 - Lab 4 - Arithmetic Circuits.....	41
5.5 - Lab 5 - Latches and Flip-Flops.....	46
5.6 - Lab 6 - Synchronous Counters	51
6 - Appendix.....	55
6.1 - Pin Assignments	55

1 - LABORATORY INSTRUCTIONS

1.1 - LABORATORY RULES

Attendance

Please be available at the beginning of the laboratory session. Lab sessions will start by taking the attendance, and then there will be a 15-20 minute introduction in which the TAs will explain the experiment. During this time you may ask your TAs any questions about the experiment. Absence from any lab session will result in a zero lab mark and the report will not be accepted.

Working group

Laboratory groups will consist of two students only. Students are required to stay in the same group and with the same TA for the whole semester. Every group performing the experiment is required to record their data on page(s) and this should be seen and signed by the TA. The data should be attached to the submitted report. One lab report is expected from each group after each lab. The lab report should be prepared according to the guidelines outlined in the next page. If you are in doubt, please ask your TA. Reports are due one week from the date of performing the experiments. Marked reports will be returned in the following lab session. Late submission of lab reports will result in a 5% deduction for each late day.

Laboratory rules

- ⚡ Do not place your coat, bags etc. on the bench.
- ⚡ Observe caution in working with equipment to avoid any risk to yourself and to your partner or cause damage to the lab or lab equipment.
- ⚡ Have your circuit checked by the TA before you switch it **ON**.
- ⚡ Clean your bench before you leave. Any trash should be put in its place.
- ⚡ Stay at your workbench and not wonder around the lab.
- ⚡ No food or drinks are permitted in the labs.

1.2 - LABORATORY REPORT FORMAT

Here is a list and brief description of the different sections a laboratory report should contain. Note that the purpose of a laboratory report is to present your results to others and, as such, should be well presented. Reports that are deemed of unacceptable quality will not be corrected.

Cover page

A cover page should include the title of the experiment, student names and numbers and the date in which the experiment was performed.

Objectives

State the main objectives and all sub-objectives. Also state any laws/rules that will be verified.

Equipment and components used

List all instruments and components used in the experiment. Clearly mention their model/type.

Circuit diagram

Draw circuit diagrams to be tested or used in the lab with all the values and component types used.

Experimental data and data processing

Using tables, show experimental data referring to procedure step number. Also, present the data in the same order as the procedure. Perform calculations using measured data. All circuit diagrams must be drawn using a ruler.

Comparison of expected data and experimental data

Perform a side-by-side comparison of expected and experimental data in a table and highlight or circle major deviations.

Discussion and conclusion

Discuss matching of expected and experimental data in general and state what laws have been verified, what sub-objectives have been met and try to find logical explanation for major deviations. Give summary of results in tabular form if necessary and state whether main objectives were met.

Laboratory data sheets

Include with the report the laboratory data sheet signed by the TA who supervised the experiment.

2 - GENERAL INTRODUCTION

2.1 - DIGITAL TEST EQUIPMENT

In order to test the various scenarios and circuits presented in this laboratory, some equipment will be needed. These will be listed and described in this section.

DE2-115 Altera card

The DE2-115 Altera card contains the Altera Cyclone IV E P4CE115 F29C7 FPGA chip on which the circuits will be programmed and tested. This will be the main tool for circuit simulation.

2.2 - COMBINATORIAL LOGIC CIRCUITS

Logic Gates

A logic gate is the simplest device used to construct digital circuits. The output voltage or logic level for each type of gate is a function of the applied input(s). Various types of logic gates are available (including inverters, ORs, NANDs, and ANDs), each with its own unique logic function. Logic circuits are constructed by interconnecting various logic gates together to implement a particular circuit function.

Truth Tables

The logic function for a single gate or a complete circuit using many gates can be easily represented in a logic truth table or a logic expression. The layout for 1 and 2 input variable truth tables is given in the following tables.

Table 2.2.1: Example truth table, two variables

A	B	Z
0	0	
0	1	
1	0	
1	1	

Input = A and B, Output = Z

Table 2.2.2: Example truth table, single variable

A	X
0	
1	

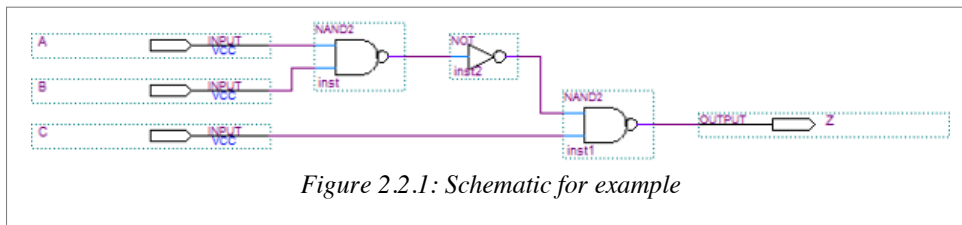
Input = A

Output = X

Example of Combinatorial Logic Circuit

Determine the logic expression for the circuit given in Figure 2.2.1 and predict the theoretical operation for that circuit. Also, show how to wire this logic circuit on a breadboard.

The schematic contains two-input NANDs and one inverter.



The logic expression for each gate output node is determined as follows:

$$X = \overline{A \cdot B}$$

$$Y = \overline{X}$$

$$Z = \overline{Y \cdot C}$$

The analysis of the circuit is performed by predicting its theoretical operation in a truth table. The function has three variables named A, B, and C. Three variables will produce a total of eight input combinations, which are listed in the truth table. Next, each of the gate output nodes are listed and then predicted, as shown in Table 2.2.3.

Table 2.2.3: Truth table prediction

A	B	C	X	Y	Z
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	1	0

The overall function of the circuit is represented by output Z for the corresponding inputs A, B, and C. Note that the resultant function for this circuit is equivalent to a 3-input NAND gate and therefore, the expression would be equivalent to:

$$Z = \overline{A \cdot B \cdot C}$$

3 - OVERVIEW OF THE ALTERA-DE2-115

This section will contain an overview of the Altera DE2-115 card. Information regarding the physical characteristics of the card, the connections or any information that is not related to software will be found in this section

3.1 - LAYOUT

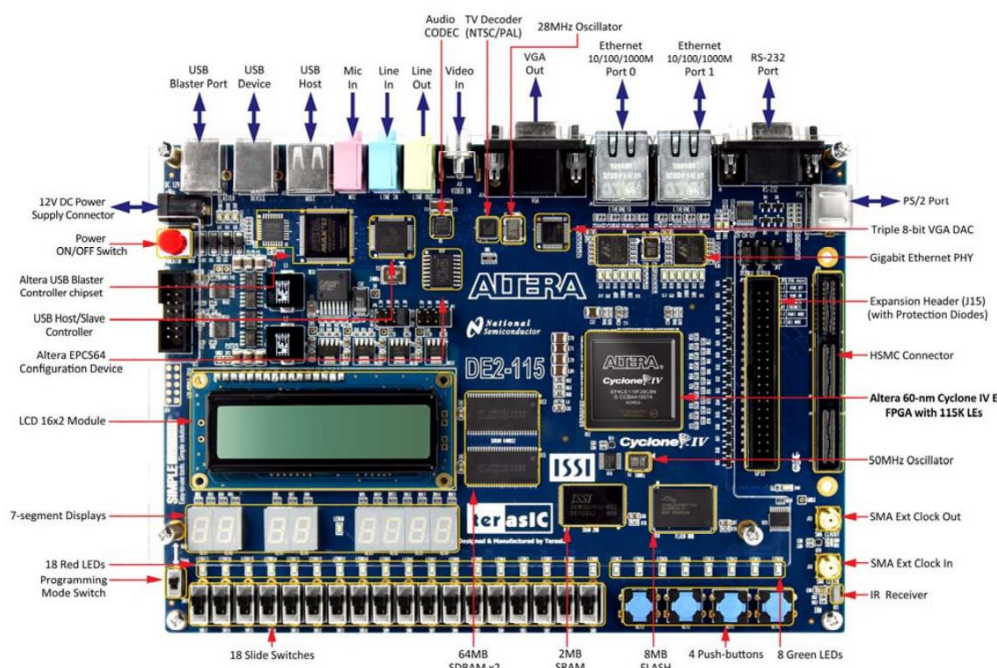


Figure 3.1.1: Altera DE2-115 card

3.2 - PHYSICAL CHARACTERISTICS

- ⤴ The Altera DE2-115 card's switches, interrupters, led and seven segment display use positive logic. This means that LEDs and 7 segment display will be activated on logic HIGH and deactivated at logic LOW. Similarly, turning a switch to the off position will generate a logic LOW.

- ⤴ However; the Altera DE2-115 cards' push buttons use negative logic. This means a button that is pushed produces a LOW signal and will produce a HIGH signal otherwise.
- ⤴ All connections are built into the board, with no need for additional wiring for basic use.
- ⤴ See Appendix for details on the switches, LEDs, 7 segment display, and the button connections of the card.

3.3 - QUICK DEBUGGING CHECK LIST

If the circuit does not function properly or use the suggestions given below to troubleshoot the malfunctioning circuit.

- ⤴ Is the Altera DE2-115 card powered?
- ⤴ When is the last time you compiled?
- ⤴ Is the correct chip (EP4CE115F29C7N) selected?
- ⤴ Is there an error in your design?

4 - INTRODUCTION TO QUARTUS II

4.1 - DOWNLOADING AND INSTALLING QUARTUS

The Quartus II 13.0 Service-Pack 1 software used in the lab is freely available on the download centre section of Altera's website (<http://dl.altera.com/13.0sp1/?edition=web>). Select the version from the drop down list at the top. (13.0 Service Pack 1 at time of writing). Download **Quartus II Software**. (Figure 4.1.1)

Design Software
Embedded Software
Archives
Licensing
Programming Software
Drivers
Board System Design
Board Layout and Test
Legacy Software

Quartus II Web Edition

Release date: June, 2013
Latest Release: v15.1

Select release: **13.0sp1**

Operating System ☒ Windows ☐ Linux
Select the operating system on which you will run the Quartus II software.

Download Method ☐ Akamai DLM3 Download Manager ☒ Direct Download
Select whether you will use the download manager (Windows only) or directly download the files.
The download manager allows you to pause the download and can help you recover from interrupted downloads.

✓ The Quartus II software version 13.0sp1 supports the following device families: Arria II, Cyclone II, Cyclone III, Cyclone IV (includes all variations), Cyclone V (includes all variations), and MAX II, MAX V, MAX 3000, MAX 7000. [More](#)

Combined Files Individual Files DVD Files

Download and install instructions: [More](#)
[Read Altera Software v13.0 Installation FAQ](#)
[Quick Start Guide](#)

Quartus II Web Edition (Free)

Quartus II Software (includes Nios II EDS) Size: 1.5 GB MD5: 70D2991B55E70EEFBBA30DB38A40BF01	Download
ModelSim-Altera Edition (includes Starter Edition) Size: 779.3 MB MD5: 97D829F95E3BDF2AD15891F00936D10	Download

Figure 4.1.1: Software and version selection

- ✓ It is important that you use version 13.0 as it is the only version that supports the DE2-115 chip set.

Sign in or sign up for myAltera, follow the instructions to begin your download. Once downloaded, run the installer.

1. Click next.
2. In the *License Agreement* page: you must read very carefully the End User Licence Agreement, agree to the terms, and click next.
3. In the *Installation directory* page: choose an Install path as well as the desired name for the program folder and click next.
4. In the *Select Components* page: Select all components for Quartus II Web Edition (Free)

Your Quartus software is now ready for use.

4.2 - CREATING A NEW PROJECT

1. Start up Quartus II 13.0 sp1 (use 64-bit edition on lab computers).
2. From the main window of the Quartus screen, access the “File” menu and choose “New Project Wizard”. This will open the project manager window.
3. Click next and enter the main directory which the project will be saved (Figure 4.2.1).
 - o Name your project something appropriate.
 - o If you are on a Lab computer, consider using either a USB drive or create a directory in your student network drive (typically Z: drive).

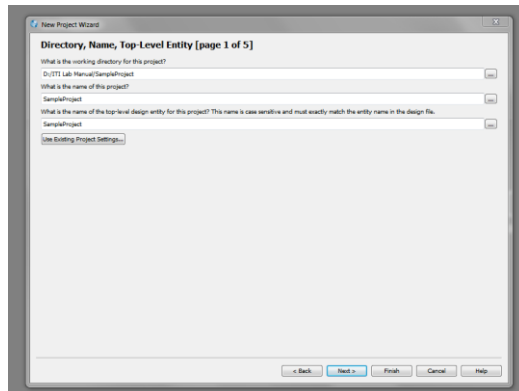
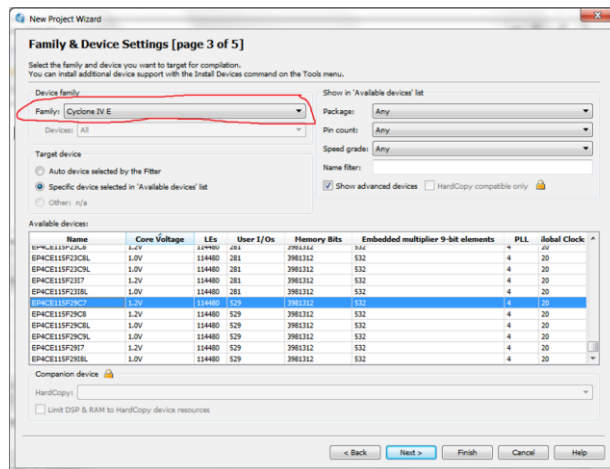


Figure 4.2.1: Creating a project from the menu

4. Click next twice (do not add other files to the project, we can do this later).
 - o At this point you should be at the “Family & Device Settings [page 3 of 5]”.
 - o Make sure the page is configured exactly like Figure 4.2.2.
 - o First select the family as **Cyclone IV E**
 - o Then select the **EP4CE115F29C7** as the device.
 - o If needed, this can be changed later in “Assignment” → “Devices”
5. Click next again and finish. Your project is now created. You should notice the project files on the left of the Quartus window.



4.3 - CREATING A NEW SCHEMATIC

Creating a new schematic file

1. From the Quartus main window, access the “File” menu and choose “New”.
2. Select “Block Diagram/Schematic” (Figure 4.3.1) and click OK.
 - o Your new schematic sheet is now displayed in the Quartus window (Figure 4.3.2).
3. Select “File” and “save as”.
4. Enter the desired name for your schematic and click Save.

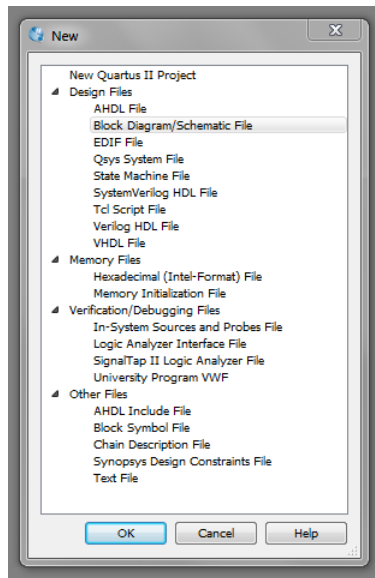


Figure 4.3.1: Creation menu

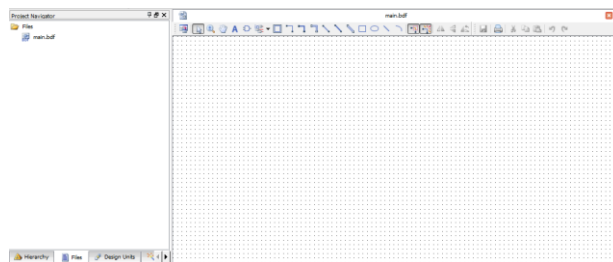


Figure 4.3.2: New schematic created

Adding components to schematic

5. Right click on the schematic sheet and select Insert then Symbol or simply click on the symbol icon. The symbol window will be displayed (Figure 4.3.3).

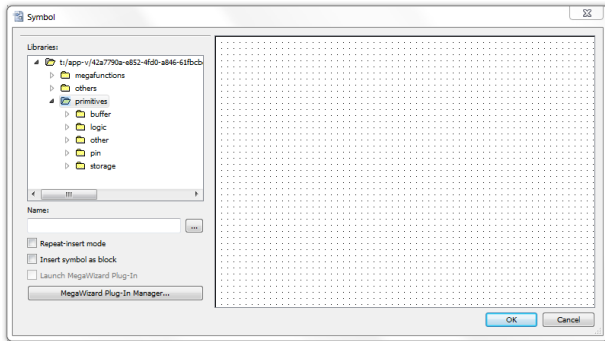


Figure 4.3.3: Add component menu

6. Navigate the “Libraries” to display the different categories of symbols (Figure 4.3.4).
 - Alternatively, the name of a symbol can be entered in the name field. Note that the name must be an exact match.

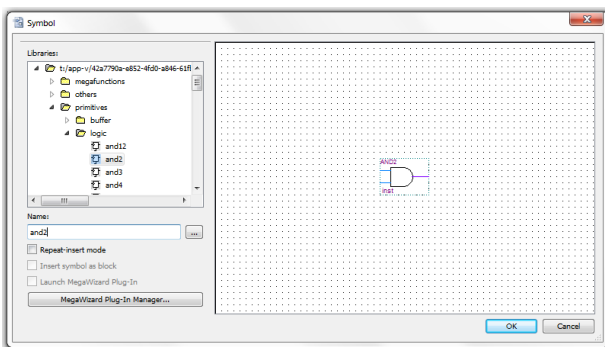


Figure 4.3.4: Component added

Common components can be found in:

- ⤴ Logic gates: /primitives/logic/
 - ⤴ Vcc and Gng: /primitives/other/
 - ⤴ Input and Output: /primitive/pin/
 - ⤴ Flip Flops: /primitive/storage/
7. When the correct component is selected, click on OK.
 - You will be brought back to the schematic sheet and the chosen gate will be attached to the cursor.
 8. Click to place component on schematic.
 9. Press the ESC key to cancel.
- ✓ It is good practice to give significant names to the inputs and outputs placed. This can be done by double clicking on the name label of components on the schematic.

Components can be dragged and dropped to arrange them as desired (Figure 4.3.5).

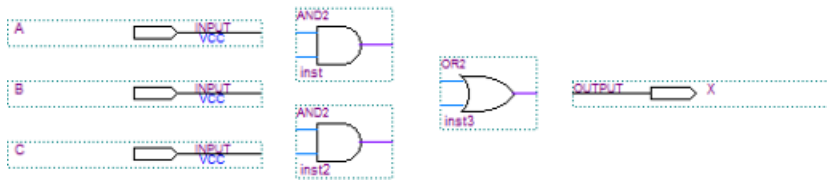


Figure 4.3.5: Three component layout

Comment [QA1]: Add input output pins to diagram. Include in diagram below (4.3.7)

Making connections

To connect the gates:


10. Click on the Orthogonal Node tool symbol  from schematic window tool bar (Figure 4.3.6)
11. Left click at the desired origin of the connection and drag to the end point (Figure 4.3.7).
 - Multiple segments of wire can be used to connect two points.



Figure 4.3.6: Schematic/block diagram tool bar

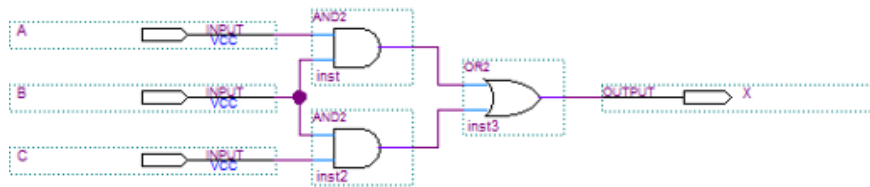


Figure 4.3.7: 3 components connected

4.4 - COMPILING THE PROJECT

Before simulating or loading projects to the DE2-115 card, they must be compiled. During compilation, Quartus will look for a “top level entry” with the same name as the project.

1. If you have named your primary file differently:
 - Go to the file tab on the left of the screen,
 - Right click on the desired file and choose “set as top level”.
2. Finally select “Start Compilation” in the Processing menu.
 - The results will be displayed in the lower side panel
 - Details can be found in the bottom dialogue box (Figure 4.4.1).

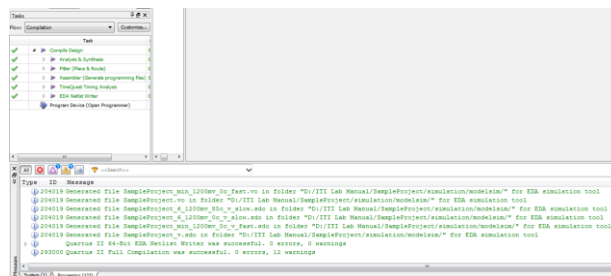


Figure 4.4.1: Compilation results

- If compilation was unsuccessful:
 - Read the error messages
 - Debug your schematic file
 - Attempting another compilation

4.5 - SIMULATING A CIRCUIT

Creating a simulation file

To simulate a circuit, a simulation file must be created:

1. Go to the file menu and click on new.
2. Select 'University Program VWF' from the list (Figure 4.5.1) and click OK.
 - o The Simulation Waveform Editor should be displayed (Figure 4.5.2)

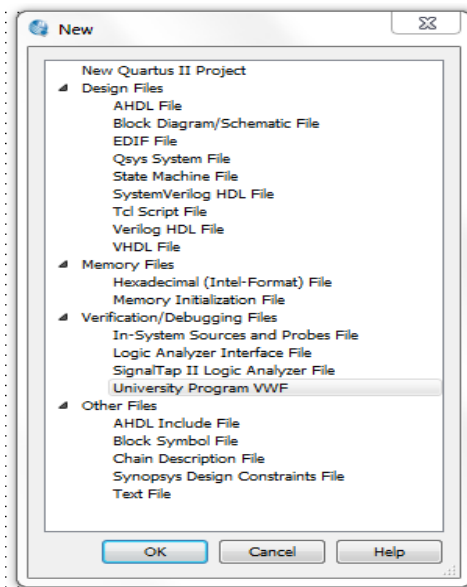


Figure 4.5.1: Creating Waveform File

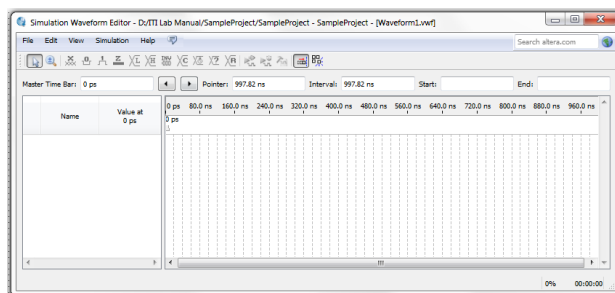


Figure 4.5.2: Waveform File

Assign nodes to simulation

Next is to assign inputs and outputs of the circuit to the simulation. To add them:

3. In the Simulation Waveform Editor right click on the left portion of the window.
4. Select Insert Node or Bus (Figure 4.5.3).
 - o A new window will appear
5. Click on “Node Finder...” (Figure 4.5.4)
6. In Node Finder, click list.
 - o Your I/O pins should appear in the left box (Figure 4.5.5)
7. Select desired pins and press the ‘>’ icon. OR select ‘>>’ to add all.
8. Click ok twice
 - o You should now be back in the Simulation Waveform Editor.

IO pins should now be available in the simulation window. (Figure 4.5.6)

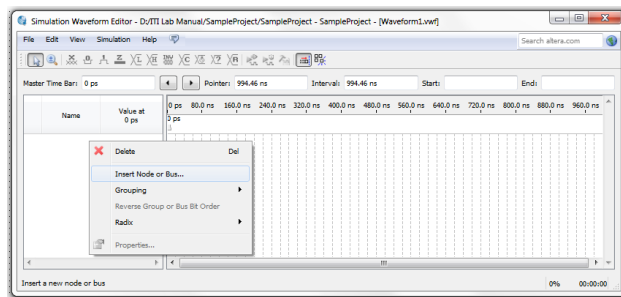


Figure 4.5.3: Accessing Node Finder

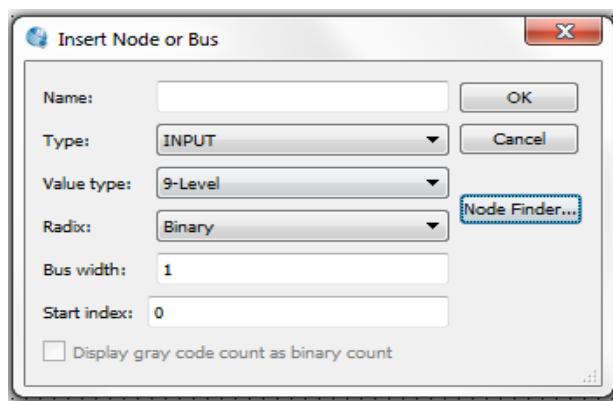


Figure 4.5.4: Search for nodes

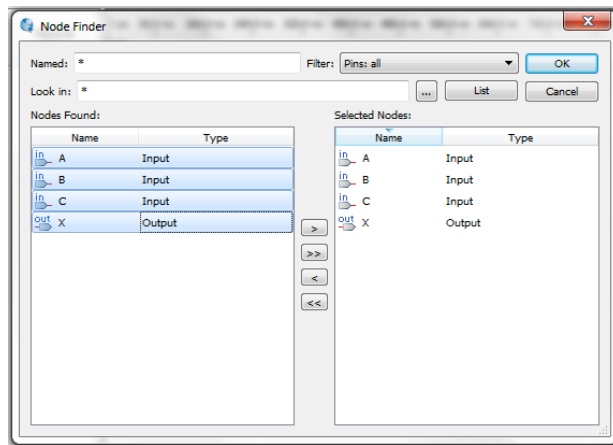


Figure 4.5.5: Adding nodes to the list

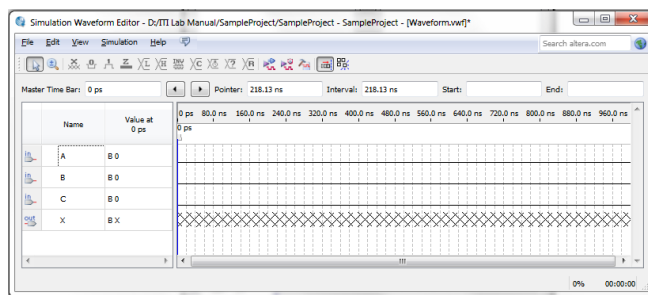



Figure 4.5.6: Nodes added

Generating Signals

In order to obtain a simulation, we are required to specify inputs. For the simulation to be complete and cover the entire input space (every possible input), all possible input combination will have been tested on the circuit.

To generate a signal:

9. In the Simulation Waveform Editor, select the desired input and choose Overwrite Clock in the icons () on the top.
 - o The window of Figure 4.5.7 will be displayed.
10. Choose a base value for the time period such as 10, 20ns, etc. and a duty cycle of 50%, and click OK
11. Repeat with all other inputs, each time doubling previous value.
 - o The inputs now have values (Figure 4.5.8).

To set maximum simulation time:

12. Go to Edit and select End Time.
13. Enter a maximum value large enough that the largest period will be displayed completely and click OK (Figure 4.5.9).
 - o You should now be back in the Simulation Waveform Editor.

Your simulation inputs are now set.

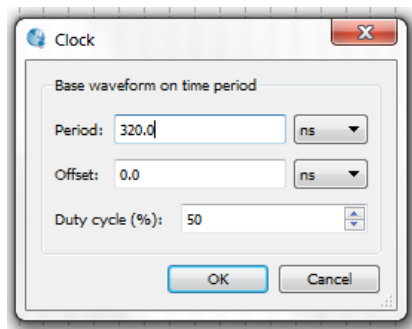


Figure 4.5.7: Overwrite Clock menu

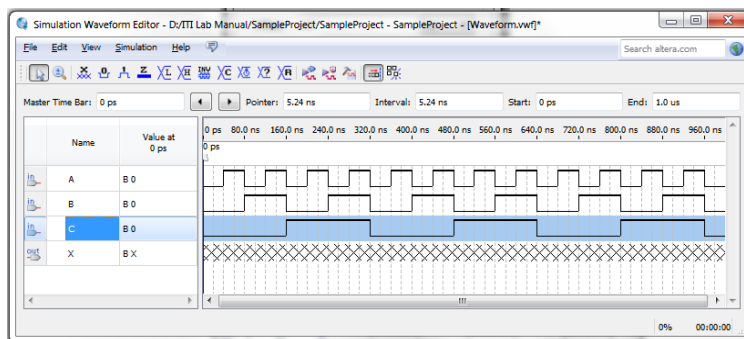


Figure 4.5.8: Input assigned

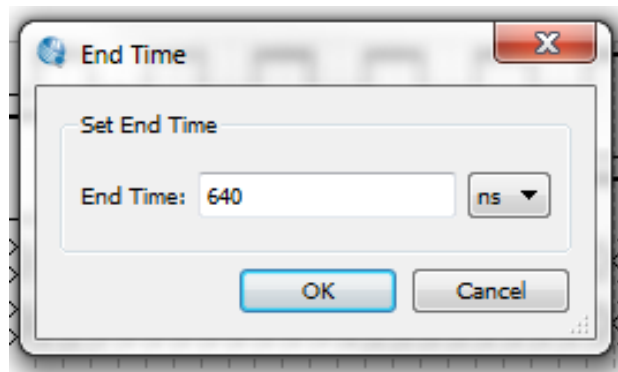


Figure 4.5.9: Setting end time

Simulating

Last stage is to simulate:

14. In the Simulation Waveform Editor, select Simulation and choose options.
15. Select “Quartus II Simulator” and click ok.
 - Ignore warnings as they are only for other boards.
16. Select Simulation again and choose Run Functional Simulation.
 - The simulation will begin
 - When completed, a report window will appear (Figure 4.5.10).

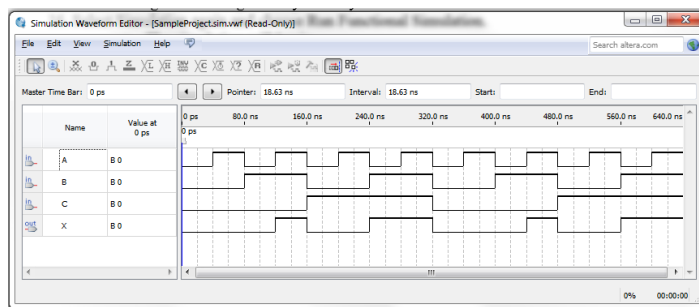


Figure 4.5.10: Simulation result

4.6 - PIN ASSIGNMENT

This sections consists in assigning the pins of the schematic to actual pins of the chip:

1. In the main window, go to Assignment and selecting “Pin Planner” (Figure 4.6.1).
 - o The pin planner window will be displayed (Figure 4.6.2).
 - o If the chip diagram does not appear, then you must set the device; see step 4 Section 4.2 - [Creating a new project](#)
2. In the pin list below the diagram, assign the appropriate pin locations to your circuit’s pins.
 - Type the pin ID into the Locations column (Ex, **PIN_G19** for the 0th red LED)
 - o Use Appendix 6.1 - [Pin Assignments](#) to determine which pins to use for IO, or use the suggested pins (according to each specific experiments).
 - o If the pins do not show, try recompiling you project.
2. Once the schematic pins have all the pins have been assigned (Figure 4.6.3), the window can be closed.
3. Recompile your project so your changes will be available for programming.

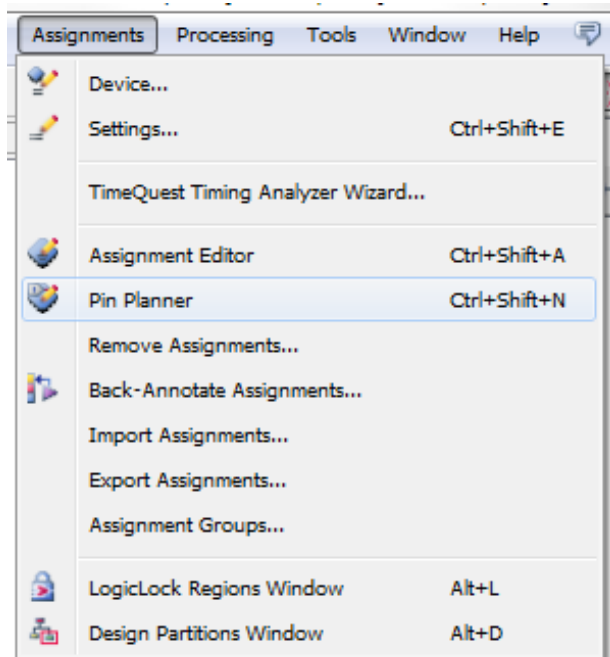


Figure 4.6.1: Accessing the pin planner

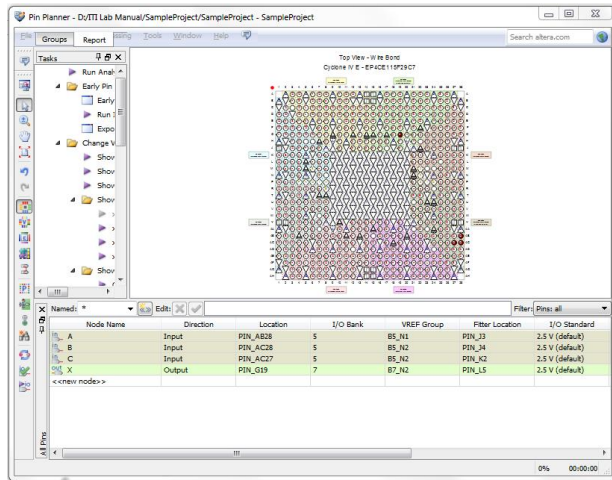


Figure 4.6.2: Pin planner window

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
A	Input	PIN_AB28	5	B5_N1	PIN_J3	2.5 V (default)
B	Input	PIN_AC28	5	B5_N2	PIN_J4	2.5 V (default)
C	Input	PIN_AC27	5	B5_N2	PIN_K2	2.5 V (default)
X	Output	PIN_G19	7	B7_N2	PIN_L5	2.5 V (default)

Figure 4.6.3: Pin assignment

4.7 - LOADING PROJECT ON THE CARD

Before uploading, check that you have completed the following:

- ✓ Assigned IO pins (see Section 4.6 - [Pin assignment](#))
- ✓ Set you main circuit as 'Top level entry' (see step 1 in Section 4.4 - [Compiling the project](#))
- ✓ Selected the correct device (see step 4 in Section 4.2 - [Creating a new project](#))
- ✓ Compiled since last changes (including any made in steps above)

To test your circuit on the DE2-115, the compiled file must be uploaded. To do so:

1. First turn on the Altera DE2-115 Board by pressing the red button near the top left corner of the board.
2. In the main Quartus window, select the Tools menu on top, select "Programmer"
 - The programmer window will open (Figure 4.7.1).
2. Ensure the device is "USB Blaster" and the "JTAG" mode is selected near top of screen (Figure 4.7.1).

- ❖ If “USB Blaster” does not appear:
 - Select Hardware Setup. A new window will appear (Figure 4.7.2)
 - Next to “Currently selected hardware” select USB-Blaster from the drop down list (Figure 4.7.2)
 - If “USB-Blaster” is not an option, close the window and reopen.
- 3. Ensure the file listed is the one you wish to load on the card.
- 4. On the Altera DE2-115 Board, set the “Programming mode switch” to “run” (bottom left of board, see Figure 3.1.1).
- 5. In the programmer window, click Start.

After the program confirms the programming has taken place, your circuit is on the card.

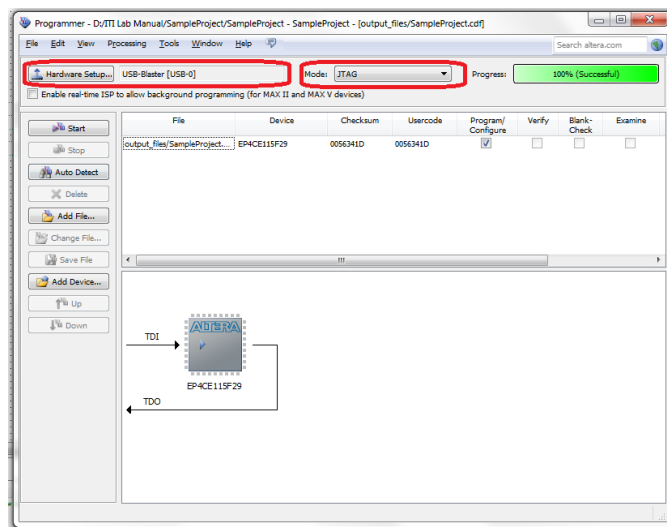


Figure 4.7.1: Accessing the programmer

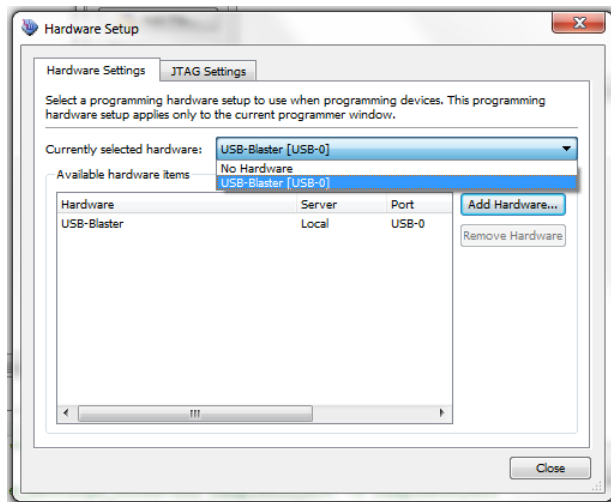


Figure 4.7.2: Programmer window

4.8 - USING 7 SEGMENT DISPLAY

The 7 segment display of the DEP-115 card is internally connected to the Cyclone IV.

To use the 7 segment display, specific pins have to be set as output:

- When assigning pins in the pin planner ensure you associate each output to the proper pin using Table 4.8.1.
 - The segments of the 7 segment display can be seen in Figure 4.8.1.
 - The decimal points on the 7 segment displays cannot be used.

Table 4.8.1: 7 segment display pins

Segment	Digit 0	Digit 1	Digit 2	Digit 3
0	PIN_G18	PIN_M24	PIN_AA25	PIN_V21
1	PIN_F22	PIN_Y22	PIN_AA26	PIN_U21
2	PIN_E17	PIN_W21	PIN_Y25	PIN_AB20
3	PIN_L26	PIN_W22	PIN_W26	PIN_AA21
4	PIN_L25	PIN_W25	PIN_Y26	PIN_AD24
5	PIN_J22	PIN_U23	PIN_W27	PIN_AF23
6	PIN_H22	PIN_U24	PIN_W28	PIN_Y19
Segment	Digit 4	Digit 5	Digit 6	Digit 7
0	PIN_AB19	PIN_AD18	PIN_AA17	PIN_AD17
1	PIN_AA19	PIN_AC18	PIN_AB16	PIN_AE17
2	PIN_AG21	PIN_AB18	PIN_AA16	PIN_AG17
3	PIN_AH21	PIN_AH19	PIN_AB17	PIN_AH17
4	PIN_AE19	PIN_AG19	PIN_AB15	PIN_AF17
5	PIN_AF19	PIN_AF18	PIN_AA15	PIN_AG18
6	PIN_AE18	PIN_AH18	PIN_AC17	PIN_AA14

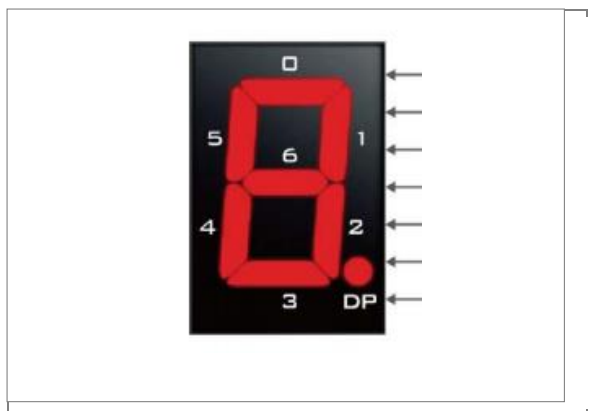


Figure 4.8.1: DE2-115, 7 segment display

4.9 - CREATING A SYMBOL FROM A CIRCUIT

It may be useful at some point to represent circuits that will be used often by symbols instead of redrawing the schematic.

To do so:

- Go to the file menu and select create/update and Create Symbol file for current file.
- The circuit can now be used as a symbol for future use.

To use new symbol:

- Go to insert symbol, a new library called 'Project' should be available
- Under 'Project' you will find your symbol

5 - LABORATORY EXPERIMENTS

5.1 - LAB 1 - LOGIC GATES

Objectives:

- Construct simple combinational logic circuits from a schematic.
- Experimentally determine the functional operation of simple combinational logic circuits.
- Identify equivalent logic gates to those produced by various circuit configurations from the resulting truth table.
- Connect various gates together to create simple logic functions.
- Analyse combinational logic circuits and predict their operation.
- Construct and test more complex combinational logic circuits.

Equipment and components

- ▲ Quartus II 13.0 Service-Pack 1
- ▲ Altera DE2-115 card

Part I – Combinational Logic Circuits Construction

Preparation

1. Write the logic expression of the circuits presented in Figure 5.1.1, Figure 5.1.2 and Figure 5.1.3.
2. Find their truth table

Procedure

1. Create a new project in Quartus. (see Section 4.2 - [Creating a new project](#))
2. Add a block diagram sheet to the project and draw the logic diagram of the circuit presented in Figure 5.1.1. (see Section 4.3 - [Creating a new schematic file](#))
3. Compile the project (see Section 4.4 - [Compiling the project](#))
4. Create a simulation and take a screen shot of the results. (see Section 4.5 - [Simulating a circuit](#))
5. Assign pins to inputs (switches) and outputs (LEDs). (see Section 4.6 - [Pin assignment](#))
6. Recompile and upload the compiled file to the Altera DE2-115 card (see Section 4.7 - [Loading project on the card](#))
7. Experimentally find out the truth table of the logic circuit loaded on the card:
 - Use each combination of the input logic variables from the truth table as the inputs to your circuit.

- Verify that the actual output of the circuit matches the corresponding output from the truth table you have theoretically derived.
- 8. Repeat step 2 to 8 for circuits of Figure 5.1.2 and Figure 5.1.3.
- 9. Identify the equivalent logic gate of each circuit

Figure 5.1.1: One-chip logic circuit

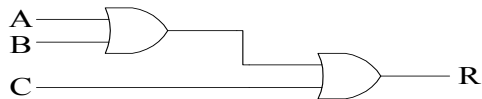


Figure 5.1.2: Two-chip logic circuit

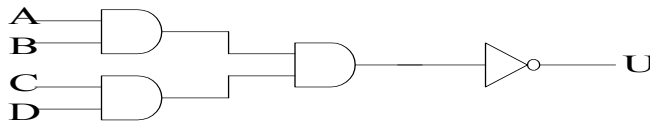
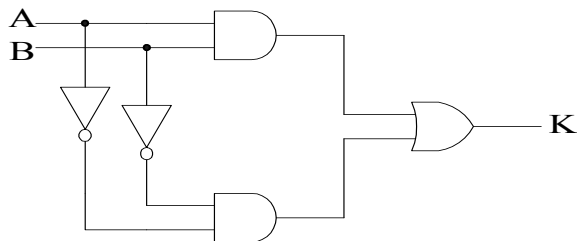


Figure 5.1.3: Three-chip logic circuit



Part II - Combinational Logic Circuits Analysis

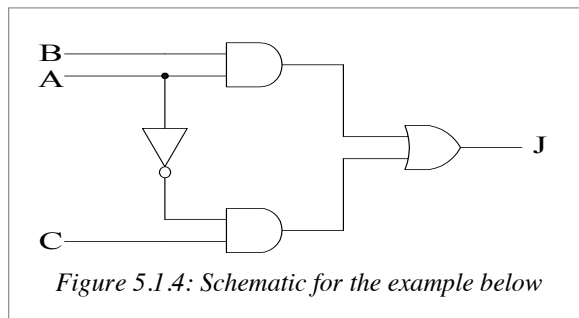
Background

The theoretical operation of a combinational logic circuit can be predicted by analysing the circuit's output for every possible input combination. The circuit analysis for each input combination is performed by determining the output of each gate, working from the input side of the circuit to the output. We will later discover shortcuts to speed up the analysis process.

Logic circuits may be functionally equivalent. They may perform the same function (i.e. their logic truth tables are identical) but be constructed from different logic gates or interconnected in an entirely different manner. In fact we will find that often a complex logic circuit can be replaced with a much simpler one that performs the identical function.

EXAMPLE OF COMBINATIONAL LOGIC ANALYSIS

Analyse the circuit in Figure 5.1.4 and determine its truth table.



The logic expression for the given logic circuit is:

$$J = AB + \bar{A}C$$

Since the logic circuit has 3 input variables, a truth table (Table 5.1.1) listing all 8 possible combinations needs to be constructed. To do so:

- Create a separate output column in the truth table for each logic gate in the circuit.
- Label the column to represent the gate function or output node name.
- Determine the output for every gate in the circuit for each row in the truth table.
- Determine the output of the overall circuit for each row in the truth table.

A	B	C	\bar{A}	AB	$\bar{A}C$	J
0	0	0	1	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	1	0	1
1	1	1	0	1	0	1

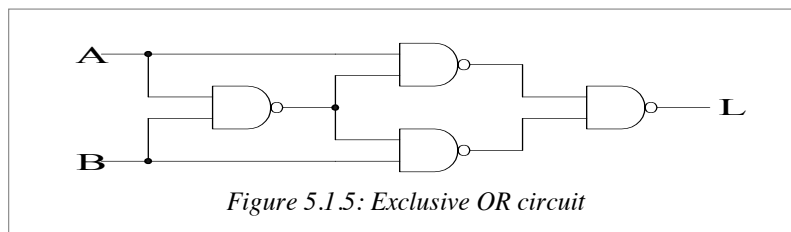
Table 5.1.1: Truth table analysis for the above example

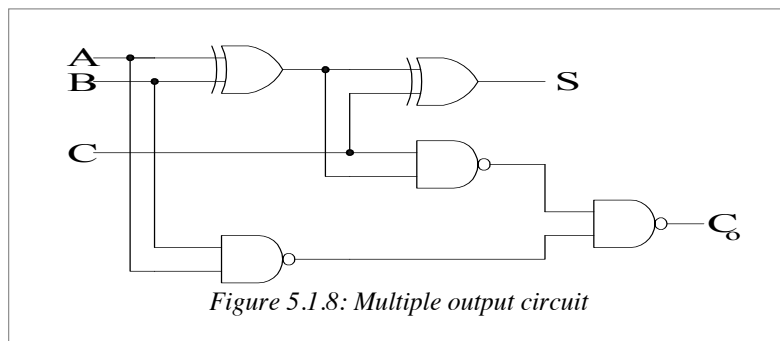
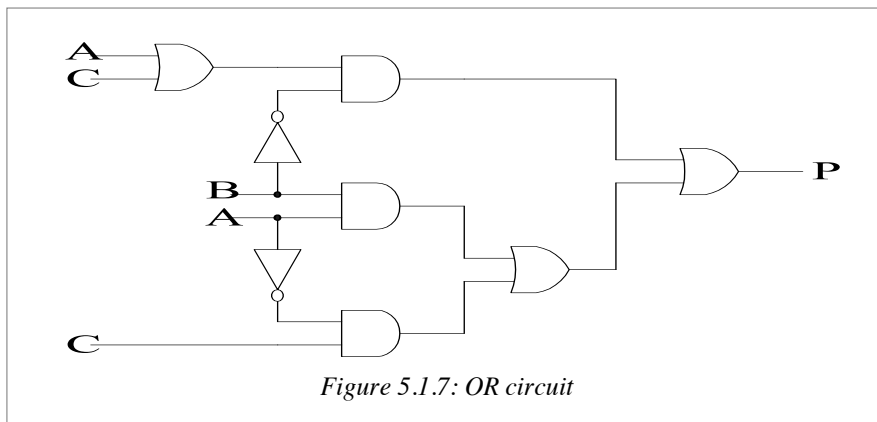
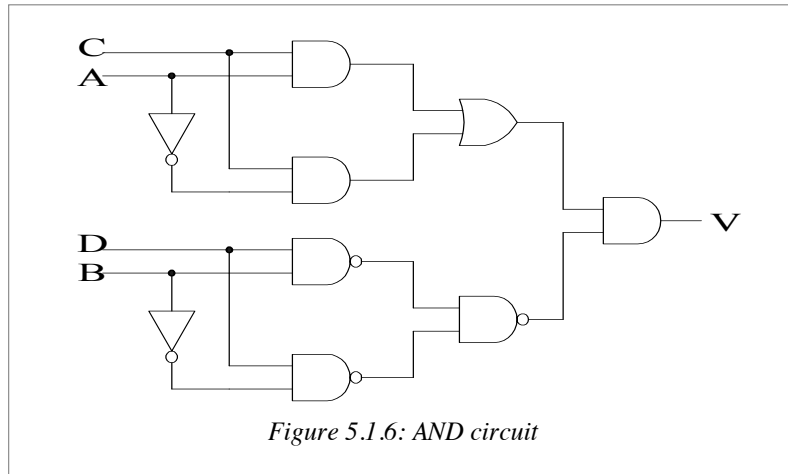
Preparation

1. Write the Boolean logic expression of the circuits in Figure 5.1.5, Figure 5.1.6, Figure 5.1.7, and Figure 5.1.8.
2. Find their truth table.

Procedure

1. Add a block diagram sheet to the project and draw the logic diagram of the circuit represented in Figure 5.1.5. (see Section 4.3 - [Creating a new schematic file](#))
2. Compile the project (see Section 4.4 - [Compiling the project](#))
3. Create a simulation and note the results. (see Section 4.5 - [Simulating a circuit](#))
4. Assign pins to inputs (switches) and outputs (LEDs). (see Section 4.6 - [Pin assignment](#))
5. Recompile and upload the compiled file to the Altera DE2-115 card (see Section 4.7 - [Loading project on the card](#))
6. Find out the truth table of the logic function that is implemented by your circuit:
 - Using each combination of the input logic variables from the truth table as the inputs to your circuit.
 - Verify that the output of the circuit matches the corresponding output from the truth table you have derived above.
7. Repeat step 1 to 7 for the circuits of Figure 5.1.6, Figure 5.1.7, and Figure 5.1.8.





5.2 - LAB 2 - BOOLEAN LOGIC

Objectives:

- ⤴ Simplify logic functions starting from their logic truth tables or Boolean expressions.
- ⤴ Synthesize, implement and test minimized combinational circuits.
- ⤴ Devise and design combinational logic circuits from specifications.
- ⤴ Implement combinational circuits using any type of available logic gates.
- ⤴ Implement combinational circuits using NAND gates only.

Equipment and components

- ⤴ Quartus II 13.0 Service-Pack 1
- ⤴ Altera DE2-115 circuit board

Part I – Combinational Logic Circuits minimization by Boolean Algebra

A logic function is given by the following Boolean expression:

$$Y = \overline{A}BCD + (\overline{A} + \overline{B})(\overline{C}D) + \overline{\overline{A} + \overline{B}}$$

Preparation

1. Draw the logic circuit that implements the above logic function.
2. Derive the truth table of Y as a function of logic variables A, B, C and D .
3. Use the rules of Boolean algebra to simplify the expression obtained in (1) as much as possible.

Procedure

1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the circuit that implements the minimised function found in pre-lab.
3. Compile the project
4. Create a simulation and note the results.
5. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
6. Recompile and upload the compiled file to the DE2-115
7. Find experimentally the truth table of your circuit:
 - Apply each combination of the logic variables A, B, C and D to its inputs.
 - Verify that the output of the minimized circuit matches the corresponding output from the truth table you initially found.

Part II – Combinational Logic Circuits minimization by the Karnaugh Map Method

Preparation

A logic function is given by the following truth table

	A	B	C	D	Y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

Table 5.2.1: Truth table of Part II logic function

1. Derive the canonical sum-of-products (SOP) expression of the function $Y(A, B, C, D)$ from Table 5.2.1.
2. Use the Karnaugh map method (Figure 5.2.1) find the simplest canonical SOP expression of the function $Y(A, B, C, D)$.

	CD	00	01	11	10
AB					
00		0	1	3	2
01		4	5	7	6
11		12	13	15	14
10		8	9	11	10

Figure 5.2.1: Karnaugh map template

3. Draw the logic diagram of the SOP minimised circuit
4. Use De Morgan's theorems to express the simplest SOP in terms of *NAND* operators.
5. Use the Karnaugh map method to find the simplest product-of-sums (POS) expression of the function $Y(A, B, C, D)$.
6. Draw the logic diagram of the circuit that implements the POS minimized expression. Analyse the circuit that implements the POS minimized expression and determine its truth table

Procedure

1. Add a block diagram sheet to the project and draw the logic diagram of the SOP minimised circuit with NAND gates.
2. Compile the project.
3. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
4. Recompile and upload the compiled file to the DE2-115.
5. Find out the truth table of the minimized circuit that you have built with NAND gates, by applying each combination of the logic variables A, B, C and D to its inputs.
6. Verify that its output matches the corresponding output from the given truth table.
7. Repeat your experiment using the logic diagram of the POS minimized circuit.
8. Show that the POS minimized circuit and SOP minimized circuit is equivalent.

Part III – Design of Combinational Logic Circuits

4 bit numbers (N) are presented to the inputs (D_3, D_2, D_1, D_0), of the logic circuit from the block diagram of Figure 5.2.2. D_3 is the most significant bit (msb) of the binary representation of N , while D_0 is the least significant bit (lsb). The output (P) of this circuit has to be *HIGH* if the current input number N is prime and *LOW* if that number is divisible.

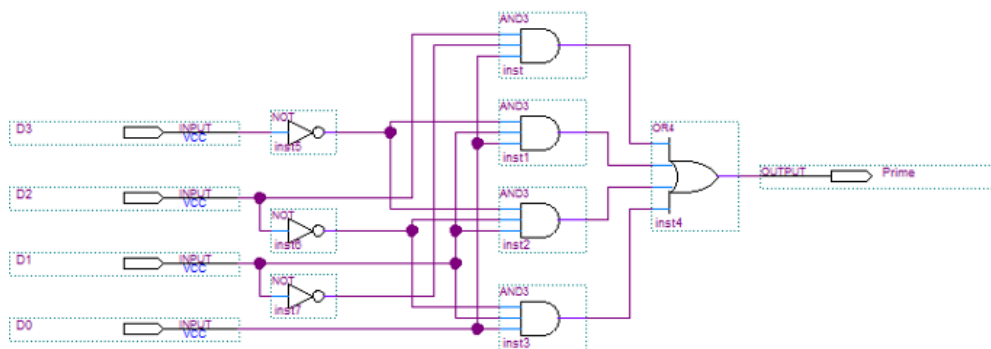


Figure 5.2.2: Block diagram of a prime number detector with AND Gates

Preparation

1. Devise and write down the truth table of P as a function of logic variables D_3 , D_2 , D_1 and D_0 using Table 5.2.2.

N	D_3	D_2	D_1	D_0	P
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

Table 5.2.2: Truth table of prime number detector

2. Derive the canonical sum-of-products (SOP) expression of the function $P(D_3, D_2, D_1, D_0)$ from its truth table.
3. Use the Karnaugh map method to reduce this canonical SOP expression to the simplest SOP form.

4. Use the De Morgan's theorems to express the simplest SOP in terms of NAND operators; implement this expression using NAND gates only.
5. Using Boolean algebra, try to minimize further the simplest SOP, at the expense of increasing the number of levels of logic gates.

Procedure

1. Add a block diagram sheet to the project and draw the logic diagram of the circuit with NAND gates.
2. Compile the project
3. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
4. Recompile and upload the compiled file to the DE2-115.
5. Experimentally find the truth table of your circuit by applying all input numbers from 0 to 15 in binary form.
6. Verify that the output of the minimized circuit matches the corresponding output from the truth table your prepared truth table.
7. Add a block diagram sheet to the project and draw the logic diagram of the minimised circuit
8. Compile the project
9. Assign pins to inputs (switches) and outputs (LEDs).
10. Recompile and upload the compiled file to the DE2-115.
11. Show that your minimized circuit is equivalent with the previous one that implemented the simplest SOP form by verifying experimentally that they have identical truth tables.

5.3 - LAB 3 - DECODERS, DISPLAYS AND MULTIPLEXERS

Objective:

- ^ Analyse, construct and test a simple 2-to-4 decoder.
- ^ Construct and test a seven-segment decoder display.
- ^ Analyse, construct and test a simple multiplexer.

Equipment and components

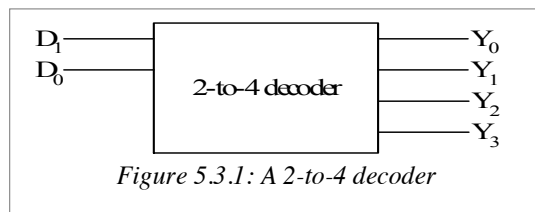
- ^ Quartus II 13.0 Service-Pack 1
- ^ Altera DE2-115 circuit board

Part I – A 2-to-4 Decoder

Background

A decoder is a combinational circuit with one or more outputs, each of which activates in response to a unique binary input value.

For example, a 2-to-4 decoder shown in Figure 5.3.1, has two inputs, D_0 and D_1 , and four outputs, Y_0 , Y_1 , Y_2 and Y_3 . Only one output is active at any time.



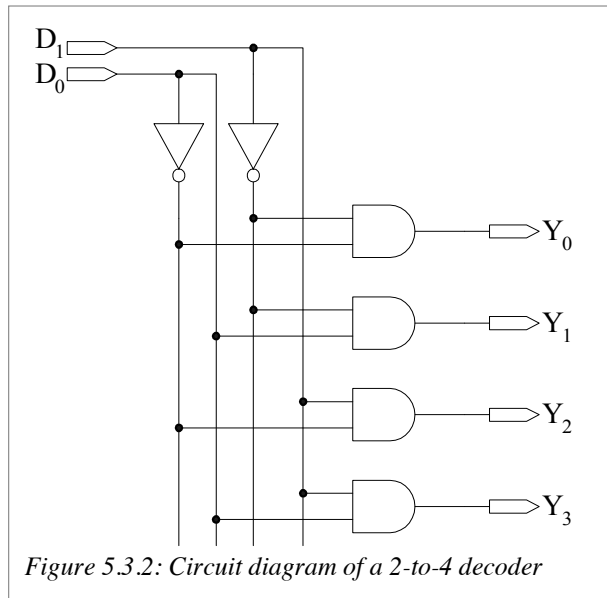
The circuit for the 2-to-4 decoder is shown below in Figure 5.3.2. Each AND gate is configured so that its output goes HIGH with a particular value of D_1D_0 . In general, the active output is the one whose subscript is equivalent to the binary value of the input.

For example, if $D_1D_0 = 10$, only the AND gate for output Y_2 has two HIGH inputs and therefore Y_2 will have a HIGH output and all others a LOW output.

Preparation

For the circuit of Figure 5.3.2,

1. Write the logic expression for each output Y_0 , Y_1 , Y_2 and Y_3 .
2. Give the truth table for each output Y_0 , Y_1 , Y_2 and Y_3 .



Procedure

1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the circuit (Figure 5.3.2).
3. Compile the project
4. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
5. Recompile and upload the compiled file to the DE2-115
6. Experimentally verify that the output of the circuit by create a truth table using all input combinations.
7. Compare the experimental truth table with your prepared truth table from pre-lab.
8. From the circuit, determine the output when $D_1D_0 = 01$.

Part II – Decoder and Seven Segment Display

Background

Decoder circuits are available for various kinds of display devices such as 7-segment displays. A 7-segment display is commonly used to display the decimal characters 0-9. The display segments are often constructed using light emitting diodes (LEDs) in which the appropriate LED segments are forward-biased (meaning LED is on) for the desired symbol shape. Decoder circuits control the LED for the display of the appropriate characters for the data being input. The pin-out configuration for a typical, 7-segment display is illustrated in Figure 5.3.3.

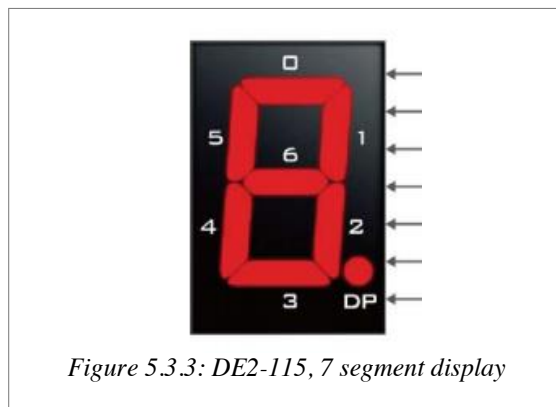


Figure 5.3.3: DE2-115, 7 segment display

Standard BCD-to-7-segment decoder chips are available to provide the necessary signals for a 7-segment LED display device to produce the decimal characters of 0 through 9. Quartus contains several standard chips that can be used to in a circuit to simulate the actual chip. For this lab, use the chip '7447' for the seven segment display.

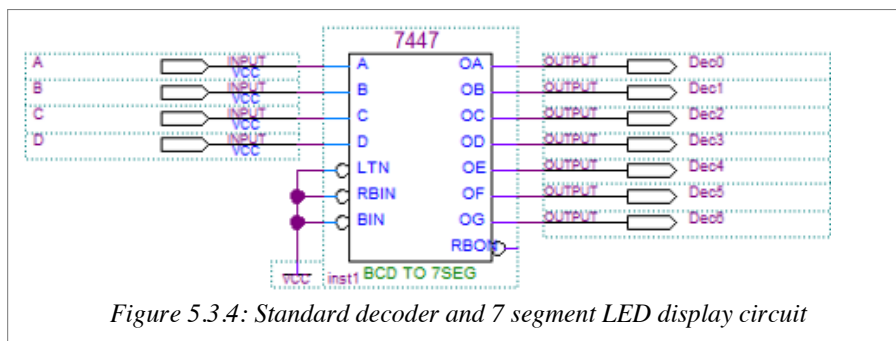


Figure 5.3.4: Standard decoder and 7 segment LED display circuit

Procedure

For the circuit in Figure 5.3.4:

1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the circuit (Figure 5.3.4).
3. Compile the project
4. Assign pins to inputs (switches) and outputs (7-Segments) (use Appendix 6.1 - , Table 6.1.4).
 - Each segment of the display has its own pin associated with it.
 - The segment is identified with numbers 0 through 6
 - For example in Figure 5.3.4, the output pin Dec0 is associated with segment 0, Dec1 is associated with segment 1, and so on.
5. Recompile and upload the compiled file to the DE2-115
6. Complete Table 5.3.1 with the display that the 7-segment LED produces for the given inputs of $D_3D_2D_1D_0$
7. Verify that the corresponding display is correct by comparing it to the truth table given in your textbook on page 174.

$D_3D_2D_1D_0$	Corresponding Display	$D_3D_2D_1D_0$	Corresponding Display
0000		0001	
0010		0011	
0100		0101	
0110		0111	
1000		1001	

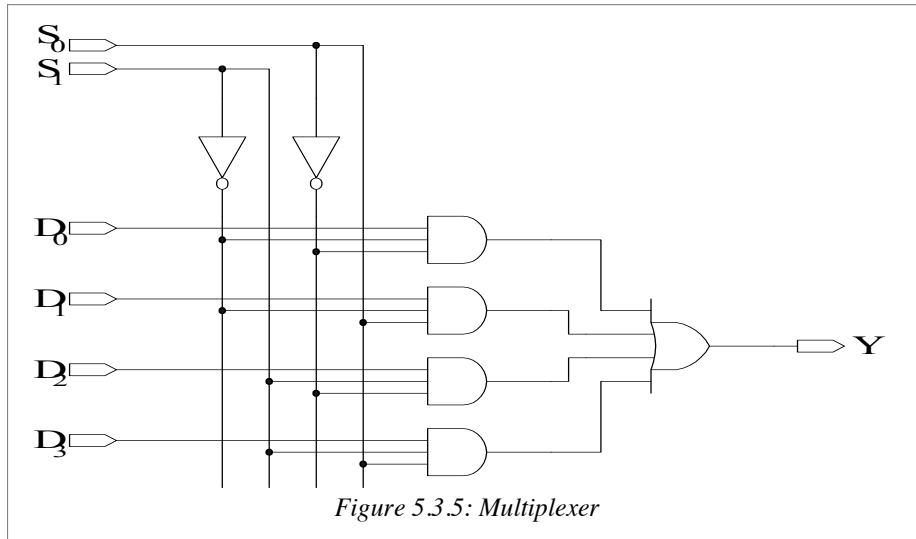
Table 5.3.1: Display corresponding to 0-9 digits

Part III – Multiplexers

Background

A multiplexer (abbreviated MUX) is a device for switching one of several digital signals to an output, under the control of another set of binary inputs. The inputs to be switched are called the *data inputs*; those that determine which signal is directed to the output are called the *select inputs*.

The multiplexer of Figure 5.3.5 has data inputs labelled D_3 to D_0 and the select inputs labelled S_1 and S_0 .



Preparation

For the circuit in Figure 5.3.5:

1. Determine the kind of multiplexer the circuit represents (i.e. is it a 2-1 MUX or a 4-1 MUX?).
2. Write the Boolean expression that describes this MUX.
3. From the Boolean expression, determine what Y is when $S_1S_0 = 01$.
4. Determine the truth table with S_1 and S_0 as the selection inputs, D_0 , D_1 , D_2 and D_3 as the data inputs and Y as the output. (using 'X' to represent don't care terms)

Procedure

1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the circuit (Figure 5.3.5).
3. Compile the project.
4. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
5. Recompile and upload the compiled file to the DE2-115
6. Obtain the experimental truth table by using each combination of the inputs from the truth table as the inputs to the circuit. Verify that the output of the circuit matches the corresponding output from the truth table obtained theoretically.

5.4 - LAB 4 - ARITHMETIC CIRCUITS

Objective:

- ⤴ Create and simulate a full adder, assign pins to the design, and test it on the Altera DE2-115 circuit board.
- ⤴ Use a full adder as a component in an 8-bit adder/subtractor.
- ⤴ Create a hierarchical design, including components for full adders and seven-segment decoders, using the QUARTUS II graphic editor.
- ⤴ Design an overflow detector for use in a two's complement adder/subtractor.

Equipment and components

- ⤴ Quartus II 13.0 Service-Pack 1
- ⤴ DE2-115 circuit board

Background

Arithmetic Circuits

Circuits for performing binary arithmetic are based on half adders, which add two bits and produce a sum and carry, and full adders, which also account for a carry added from the lesser-significant bit.

- Full adders can be grouped together to make a parallel binary adder, with n full adders allowing two n -bit numbers to be added, generating an n -bit sum and a carry output.
- A parallel adder can be converted to a two's complement adder/subtractor by including XOR functions on the inputs of one set of operand bits, say input B, allowing the operations $A + B$ or $A - B$ to be performed.
- A control input, SUB (for SUBtract):

Table 5.4.1: Adder function table

SUB	C IN0	Math Equation	Description
0	0	$A+B$	Addition
0	1	$A+B+1$	Addition plus 1
1	0	$(A + \overline{B} = A - B - 1)$	One's Complement Subtraction
1	1	$(A + \overline{B} + 1 = A - B)$	Subtraction

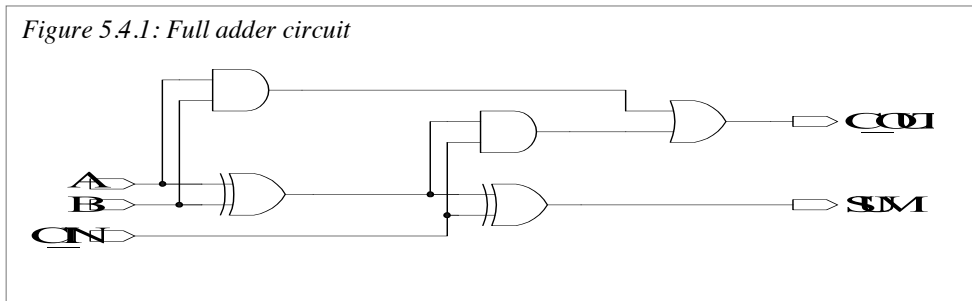
- If HIGH it causes the XORs to invert the B bits, producing the ones complement of B.

- If HIGH and the carry is HIGH then the result is $(A + \bar{B} + 1 = A - B)$
- If LOW, it transfers B to the parallel adder without modification.
- If LOW, and the carry input is HIGH then the result is B+1
- Sign-bit overflow occurs when a two's complement sum of difference exceeds the permissible range of numbers for a given bit size.
 - This can be detected by an SOP circuit that compares the operands and result sign bits of a parallel adder if the 2's complement is used to represent the.

Part I – Full Adders

Procedure

Figure 5.4.1: Full adder circuit



1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the circuit represented in Figure 5.4.1.
3. Compile the project
4. Create and run a simulation for the full adder and show it to your instructor.
5. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
6. Recompile and upload the compiled file to the DE2-115
7. Take the experimental truth table of the full adder to verify its operation. Show the results to your instructor.

Part II – Parallel Adder

Preparation

1. In Table 5.4.2, giving the expected sum in both binary and hexadecimal. Show the carry output separately. Below are examples of addition with separate carry output.

e.g.: $10111111 + 10000001 = 01000000$ (Carry output = 1)

Hexadecimal equivalent: $(BF)_{16} + (81)_{16} = (40)_{16}$ (Carry output = 1)

e.g.: $00111111 + 00000001 = 01000000$ (Carry output = 0)

Hexadecimal equivalent: $(3F)_{16} + (01)_{16} = (40)_{16}$ (Carry output = 0)

Procedure

1. Use the circuit of part I to create a full adder block (see Section 4.9 - [Creating a symbol from a circuit](#)).
2. Add a block diagram sheet to the project and draw the logic diagram of an 8-bit parallel adder using the full adder block.
3. Simulate your circuit with the inputs of Table 5.4.2 and compare your results. Show your simulation to your instructor.
4. Compile the project
5. For inputs and outputs (use Appendix 6.1 -):
 - Assign input pins to switches
 - Assign numerical output pins to red LEDs.
 - Assign carry output pins to green a LED.
 - Ensure that your input switches and output LEDs are in order of decreasing significance.
6. Recompile and upload the compiled file to the DE2-115
7. Test the operation of the 8-bit parallel adder by applying the combinations of inputs A and B listed in Table 5.4.2. Show the results to your instructor.

Table 5.4.2: Sample Sums for an 8-bit parallel adder

Binary Inputs	Carry	Binary Sum	Hex Equivalent
$01111111 + 00000001$			
$11111111 + 00000001$			
$11000000 + 01000000$			
$11000000 + 10000000$			

Part III – Two's Complement Adder/Subtractor (Optional)

Preparation

1. Follow the examples shown to add the signed binary numbers in Table 5.4.3, giving the sum in both binary and hexadecimal. Show the carry and overflow outputs separately from the sum.
 - Carry output maintains the same functionality as before, and does not need to be modified for signed arithmetic.

e.g.: $01111111 + 01000001 = 11000000$ (Overflow = 1, Carry output = 0);

Hexadecimal equivalent: $(7F)_{16} + (41)_{16} = (C0)_{16}$

e.g.: $00000000 - 00000011 = 11111101$ (Overflow = 0, Carry output = 0);

Hexadecimal equivalent: $(00)_{16} - (03)_{16} = (FD)_{16}$

2. (two's complement: $(FD)_{16} = 11111101 = (-3)_{10}$)

Procedure

1. Modify the 8-bit adder you created to make an 8-bit two's complement adder/subtractor.
2. Include an overflow detector for when the output of the adder/ subtractor overflows beyond the permissible range of the values for an 8-bit (2's complement) signed number.
3. Compile the project
4. For inputs and outputs (use Appendix 6.1 -):
 - Assign pins same as for the 8-bit adder.
 - Assign Add/Sub input to a switch.
 - Assign overflow output to a **green** LED.
5. Recompile and upload the compiled file to the DE2-115
6. Test the operation of the circuit by applying the A and B inputs from Table 5.4.3. Show the results to your instructor.

Table 5.4.3: Sample sums and differences for an 8-bit adder/subtractor

Binary Inputs	Overflow	Carry	Binary Sum	Hex Equivalent	Two's Compl.
01111111 + 00000001					
11111111 + 00000001					
00000000 - 00000001					
00000000 - 01111111					
11000000 + 01000000					
11000000 + 10000000					

5.5 - LAB 5 - LATCHES AND FLIP-FLOPS

Objectives:

- ⤴ Provide insight into the characteristics of several important latches and flip-flops.
- ⤴ Build latches and flip-flops from basic gates.
- ⤴ Explain concepts of latching and edge-triggering.
- ⤴ Test latches and flip-flops to understand their operation

Equipment and components

- ⤴ Quartus II 13.0 Service-Pack 1
- ⤴ Altera DE2-115 circuit board

Background

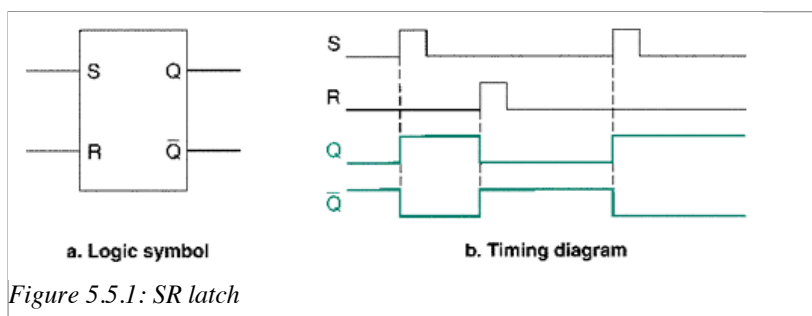
A sequential circuit is a digital circuit whose output depends not only on the present combination of input, but also on the history of the circuit.

A sequential circuit element is a basic memory element that can store 1 bit. There are two basic types of sequential circuit elements: the latch and the flip-flop. The difference is the condition under which the stored bit changes.

Part I – SR Latch

Background

The simplest sequential circuit element is the SR latch. It presents two asynchronous inputs (SET –that makes the device store a logic 1 and RESET – an input that makes the device store a logic 0) and two complementary outputs, Q and \bar{Q} that are always in opposite logic states (Figure 5.5.1)



The SR latch presents two stable states:

- ⤴ SET or ON when $Q = 1$ and $\bar{Q} = 0$.

⚠ RESET or OFF when $Q = 0$ and $\bar{Q} = 1$.

The four possible input combinations will generate the following actions of the latch:

S	R	Action
0	0	Output does not change from the previous state
0	1	RESET
1	0	SET
1	1	Forbidden condition: output depends on implementation of SR latch

Table 5.5.1: SR latch truth table

A SR latch can be implemented both with NOR (Figure 5.5.2) or NAND gates

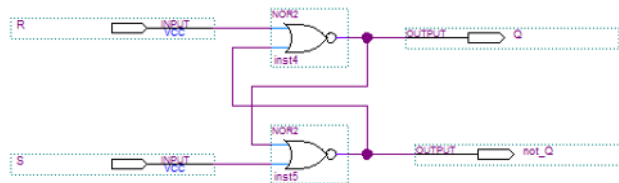


Figure 5.5.2: SR latch implemented with NOR gates

Procedure

1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the NAND SR latch
3. Compile the project
4. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
5. Recompile and upload the compiled file to the DE2-115
6. Experimentally find out the function table of the SR latch that you have built, i.e., $Q_{t+1} = f(S, R, Q_t)$ by applying each allowed combination to its inputs.

Part II – D Latch

Background

The forbidden state is eliminated in the *D latch* (Figure 5.5.3). This latch has two operating modes that are controlled by the ENABLE input (*EN*): when the *EN* is active, the latch output follows the data input (*D*) and when *EN* is inactive, the latch stores the data that was present when *EN* was last active.

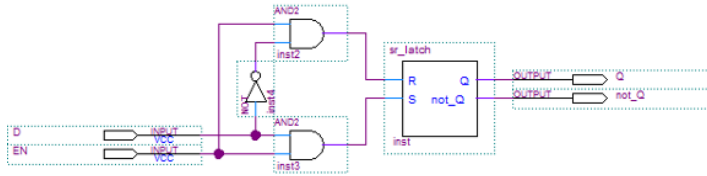


Figure 5.5.3: D latch implemented with a SR latch

The *D latch* behaviour is presented in the following function table:

EN	D	Q_{t+1}	\overline{Q}_{t+1}	Action	Comment
0	x	Q_t	\overline{Q}_t	Output does not change from the previous state	Store
1	0	0	1	RESET	
1	1	1	0	SET	

Table 5.5.2: Truth table of type D latch

Procedure

1. Create a new project in Quartus
2. Add a block diagram sheet to the project and draw the logic diagram of the NAND D-latch.
 - o You will first need to create a block diagram of your SR latch.
3. Compile the project
4. Assign pins to inputs (switches) and outputs (LEDs) (use Appendix 6.1 -).
5. Recompile and upload the compiled file to the DE2-115
6. Find out the function table of the *D latch* that you have built, and compare it with the one given above.
7. Experimentally show that the latch output follows the *D* input when *EN* is HIGH, but doesn't change with *D* as the *EN* is LOW.

Part III – D Flip-Flop

Background

A Flip-Flop is a gated latch with a clock input. The flip-flop output changes when its CLOCK input (*CLK*) detects an edge. This sequential circuit element is edge-sensitive (and not level-sensitive, as the *latch*).

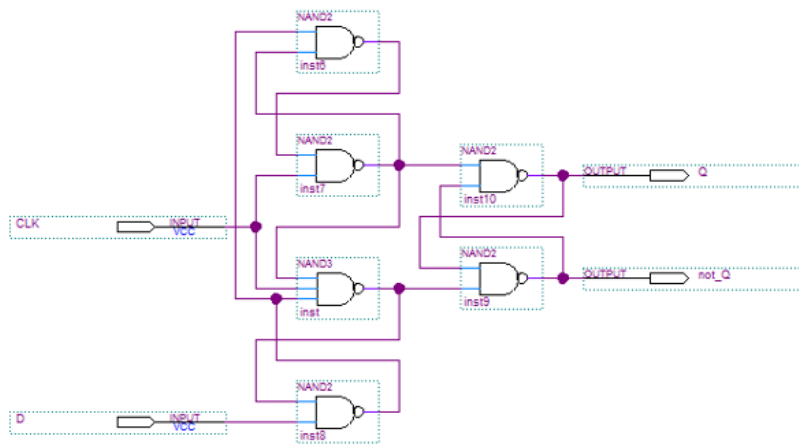


Figure 5.5.4: D flip-flop

Table 5.5.3 shows that the D input is sampled during the positive edge of the clock signal (CLK) and it is stored in the D Flip-Flop and presented to its output. Any other changes that may occur at the both CLK and D inputs, while the CLK input is stable (at 0 or 1) or it is down going, will have no effect to the flip-flop state and its output.

CLK	D	Q_{t+1}	\overline{Q}_{t+1}	Function
\uparrow	0	0	1	RESET
\uparrow	1	1	0	SET
0	X	Q_t	\overline{Q}_t	Inhibited
1	X	Q_t	\overline{Q}_t	Inhibited

Table 5.5.3: Truth table of positive edge triggered type D latch

Procedure

1. Add a block diagram sheet to the project and draw the block diagram of the D flip-flop from Figure 5.5.4.
2. Compile the project
3. For inputs and outputs (use Appendix 6.1 -).
 - Assign D to a switch
 - Assign CLK to a push button
 - Assign output to any LED
4. Recompile and upload the compiled file to the DE2-115
5. Experimentally determine on which edge of the CLK signal triggering occurs.
 - Recall that the push buttons operate on negative logic

- Make note in your lab report the procedures you have used.
- 2. Experimentally find out the function table of the *D flip-flop* that you have built, and compare it with the one given above.

Part IV – T Flip-Flop

Background

A T Flip-Flop is a flip-flop whose output toggles between HIGH and LOW on each clock pulse when input *T* is active. A T flip-flop can be constructed using a D flip-flop.

Procedure

1. Add a block diagram sheet to the project and draw the logic diagram of a T flip-flop using D flip-flop from Part III.
 - You will need to create a block diagram of your D flip-flop.
2. Compile the project.
3. For inputs and outputs (use Appendix 6.1 -).
 - Assign T to a switch
 - Assign CLK to a push button
 - Assign output to any LED
4. Recompile and upload the compiled file to the DE2-115
5. Experimentally find out the excitation table of the built T flip-flop.

5.6 - LAB 6 - SYNCHRONOUS COUNTERS

Objectives:

Upon completion of this laboratory exercise, you should be able to:

- ⤴ Design synchronous counters
- ⤴ Simulate the functions of the various counters in this laboratory exercise.
- ⤴ Display counter outputs as binary values on LEDs and test these counters.

Equipment and components

- ⤴ Quartus II 13.0 Service-Pack 1
- ⤴ Altera DE2-115 circuit board

Design Notes

The count sequence is the starting point in the design of synchronous counters; it can be specified by a table or a state diagram.

The following is an example of the design process of a synchronous counter.

- The count sequence given by the state diagram of Figure 5.6.1.

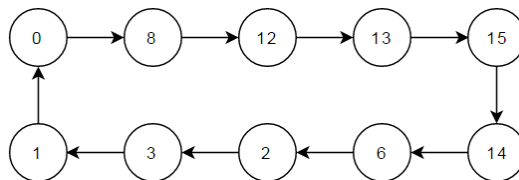


Figure 5.6.1: State diagram of count sequence

Since the state codes are represented by 4 bits, there are required 4 flip-flops to implement the synchronous counter.

Based on this state diagram, the counter's state table is derived.

- The first 2 groups (each of 4 columns) of Table 5.6.1 describe the present and the next state of the counter.
 - The next 4 groups (each of 2 columns) are the inputs of the 4 flip-flops (Q_3 , Q_2 , Q_1 , Q_0).
- ✓ While synchronous counters can be designed using any type of flip-flop, JK and D flip-flops are commonly used for this function; our example is making use of JK flip-flops.

The corresponding JK inputs are derived for every transition from the excitation table of the JK flip-flop.

Present state ⁽ⁿ⁾				Next state ⁽ⁿ⁺¹⁾				Q ₃ input		Q ₂ input		Q ₁ input		Q ₀ input	
Q ₃	Q ₂	Q ₁	Q ₀	Q ₃	Q ₂	Q ₁	Q ₀	J ₃	K ₃	J ₂	K ₂	J ₁	K ₁	J ₀	K ₀
0	0	0	0	1	0	0	0	1	x	0	x	0	x	0	x
0	0	0	1	0	0	0	0	0	x	0	x	0	x	x	1
0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
0	0	1	1	0	0	0	1	0	x	0	x	x	1	x	0
0	1	0	0	x	x	x	x	x	x	x	x	x	x	x	x
0	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x
0	1	1	0	0	0	1	0	0	x	x	1	x	0	0	x
0	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x
1	0	0	0	1	1	0	0	x	0	1	x	0	x	0	x
1	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x
1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x
1	1	0	0	1	1	0	1	x	0	x	0	0	x	1	x
1	1	0	1	1	1	1	1	x	0	x	0	1	x	x	0
1	1	1	0	0	1	1	0	x	1	x	0	x	0	0	x
1	1	1	1	1	1	1	0	x	0	x	0	x	0	x	1

Table 5.6.1: Truth table of counter

The equations of the synchronous inputs are found from the corresponding Karnaugh maps.

- Don't care terms are marked with x's
- Unused states are marked with bold x's

		Q ₁ Q ₀ 00 01 11 10						Q ₁ Q ₀ 00 01 11 10			
Q ₃ Q ₂	0	1	0	0	0	Q ₃ Q ₂	00	x	x	x	x
	01	x	x	x	0		01	x	x	x	x
	11	x	x	x	x		11	0	0	0	1
	10	x	x	x	x		10	0	x	x	x
$J_3 = \overline{Q_1} \cdot \overline{Q_0}$						$K_3 = Q_1 \cdot \overline{Q_0}$					

Table 5.6.2: JK Flip-flop 3 Karnaugh map

Q ₁ Q ₀ 00 01 11 10		Q ₁ Q ₀ 00 01 11 10	
Q ₃ Q ₂		Q ₃ Q ₂	
00	0 0 0 0	00	x x x x
01	x x x x	01	x x x 1
11	x x x x	11	0 0 0 0
10	1 x x x	10	x x x x

$J_2 = Q_3$ $K_2 = \overline{Q_3}$

Table 5.6.3: JK Flip-flop 2 Karnaugh maps

Q ₁ Q ₀ 00 01 11 10		Q ₁ Q ₀ 00 01 11 10	
Q ₃ Q ₂		Q ₃ Q ₂	
00	0 0 x x	00	x x 1 0
01	x x x x	01	x x x 0
11	0 1 x x	11	x x 0 0
10	0 x x x	10	x x x x

$J_1 = Q_3 \cdot Q_0$ $K_1 = \overline{Q_3} \cdot Q_0$

Table 5.6.4: JK Flip-flop 1 Karnaugh map

Q ₁ Q ₀ 00 01 11 10		Q ₁ Q ₀ 00 01 11 10	
Q ₃ Q ₂		Q ₃ Q ₂	
00	0 x x 1	00	x 1 0 x
01	x x x 0	01	x x x x
11	1 x x 0	11	x 0 1 x
10	0 x x x	10	x x x x

$J_0 = Q_2 \text{ xor } Q_1$ $J_1 = \overline{Q_3} \text{ xor } Q_1$

Table 5.6.5: JK Flip-flop 0 Karnaugh map

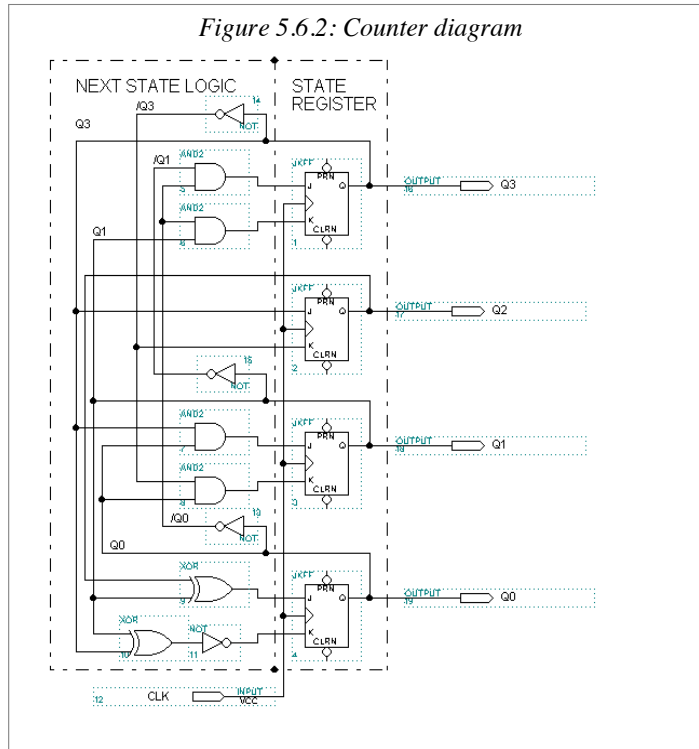
4-bit Synchronous Counter

Procedure

1. Create a new Quartus project
2. Add a block diagram sheet to the project and draw the logic diagram of the 4 bit synchronous counter given in Figure 5.6.2.
3. Compile the project
4. Simulate your circuit
4. Assign pins to inputs (button) and outputs (LEDs) (use Appendix 6.1 -).

5. Upload the compiled file to the DE2-115
6. Find experimentally the count table of your synchronous counter by pressing the pushbutton until you rollover a full counting sequence.
7. Verify that the output of your synchronous counter matches the corresponding the state diagram you were initially given. Demonstrate the operation of your circuit to your instructor.

Figure 5.6.2: Counter diagram



6 - APPENDIX

6.1 - PIN ASSIGNMENTS

Table 6.1.1: Pin assignments for LED outputs

LED Name	FPGA Pin ID		LED Name	FPGA Pin ID
LEDR0	PIN_G19		LEDR9	PIN_G17
LEDR1	PIN_F19		LEDR10	PIN_J15
LEDR2	PIN_E19		LEDR11	PIN_H16
LEDR3	PIN_F21		LEDR12	PIN_J16
LEDR4	PIN_F18		LEDR13	PIN_H17
LEDR5	PIN_E18		LEDR14	PIN_F15
LEDR6	PIN_J19		LEDR15	PIN_G15
LEDR7	PIN_H19		LEDR16	PIN_G16
LEDR8	PIN_J17		LEDR17	PIN_H15
LEDG0	PIN_E21		LEDG5	PIN_G20
LEDG1	PIN_E22		LEDG6	PIN_G22
LEDG2	PIN_E25		LEDG7	PIN_G21
LEDG3	PIN_E24		LEDG8	PIN_F17
LEDG4	PIN_H21			

Table 6.1.2: Pin assignments for slide switches

Switch Name	FPGA Pin ID	Switch Name	FPGA Pin ID
SW0	PIN_AB28	SW9	PIN_AB25
SW1	PIN_AC28	SW10	PIN_AC24
SW2	PIN_AC27	SW11	PIN_AB24
SW3	PIN_AD27	SW12	PIN_AB23
SW4	PIN_AB27	SW13	PIN_AA24
SW5	PIN_AC26	SW14	PIN_AA23
SW6	PIN_AD26	SW15	PIN_AA22
SW7	PIN_AB26	SW16	PIN_Y24
SW8	PIN_AC25	SW17	PIN_Y23

Table 6.1.4: Pin assignments for push buttons

Button Name	FPGA Pin ID	Button Name	FPGA Pin ID
KEY0	PIN_M23	KEY2	PIN_N21
KEY1	PIN_M21	KEY3	PIN_R24

Table 6.1.3: 7-segment display pins

Segment	Digit 0	Digit 1	Digit 2	Digit 3
0	PIN_G18	PIN_M24	PIN_AA25	PIN_V21
1	PIN_F22	PIN_Y22	PIN_AA26	PIN_U21
2	PIN_E17	PIN_W21	PIN_Y25	PIN_AB20
3	PIN_L26	PIN_W22	PIN_W26	PIN_AA21
4	PIN_L25	PIN_W25	PIN_Y26	PIN_AD24
5	PIN_J22	PIN_U23	PIN_W27	PIN_AF23
6	PIN_H22	PIN_U24	PIN_W28	PIN_Y19
Segment	Digit 4	Digit 5	Digit 6	Digit 7
0	PIN_AB19	PIN_AD18	PIN_AA17	PIN_AD17
1	PIN_AA19	PIN_AC18	PIN_AB16	PIN_AE17
2	PIN_AG21	PIN_AB18	PIN_AA16	PIN_AG17
3	PIN_AH21	PIN_AH19	PIN_AB17	PIN_AH17
4	PIN_AE19	PIN_AG19	PIN_AB15	PIN_AF17
5	PIN_AF19	PIN_AF18	PIN_AA15	PIN_AG18
6	PIN_AE18	PIN_AH18	PIN_AC17	PIN_AA14